Basic Division Scheme

台灣大學電子所

吳安宇 博士

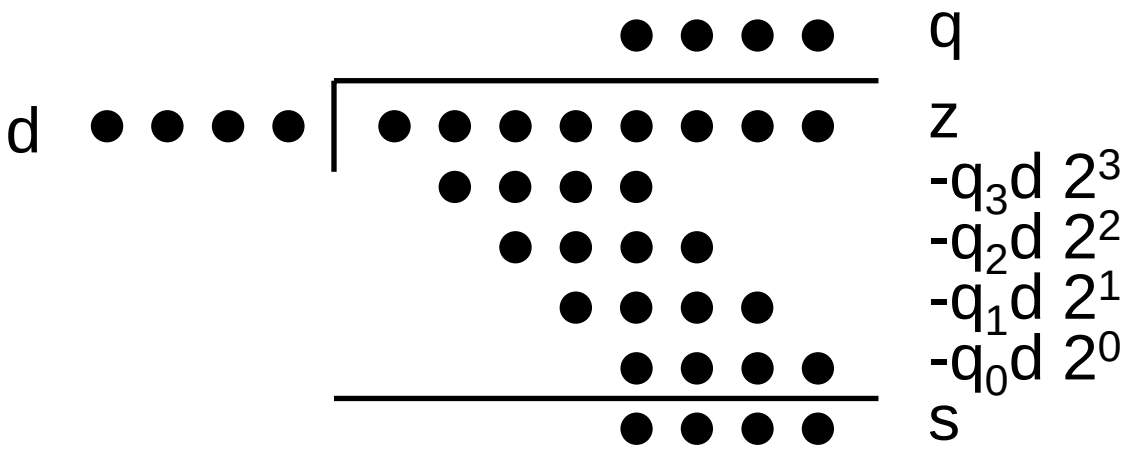
2002

Outline

- Shift/subtract division algorithm.
- Programmed division.
- Restoring hardware dividers.
- Nonstoring and signed division.
- Radix-2 SRT divisioin.
- High-Radix division.

Shift/Subtract Division Algorithms

z	*Dividend*	$z_{2k-1}z_{2k-2}...z_1z_0$
d	*Divisor*	$d_{k-1}d_{k-2}...d_1d_0$
q	*Quotient*	$q_{k-1}q_{k-2}...q_1q_0$
s	*Remainder*$[z-(d\times q)]$	$s_{k-1}s_{k-2}...s_1s_0$



Division can be done by a sequence of shifts and subtract.

Overflow Check

- Multiplication:
  - product of two *k*-bit numbers is always 2*k* bits.

integer

$$z = [(d \times q)] + s \quad integer$$

$$z < (2^k - 1)d + d = 2^k d$$

- Division:
  - quotient of a 2*k*-bit number divided by a *k*-bit number may more than *k* bits.

fractions

$$2^{-2k} z = [(2^{-k} d) \times (2^{-k} q)] + 2^{-2k} s \quad fractions$$

$$z_{frac} = [(d_{frac} \times q_{frac})] + 2^{-k} s_{frac}$$

$$Z_{frac} < d_{frac}$$

Sequential Division Algorithm

- Left shift *partial remainder*, align to the term to be subtracted.

$$S^{(j)} = 2s^{(j-1)} - q_{k-j}(2^k d) \quad with \quad s^{(0)} = z \quad and \quad s^{(k)} = 2^k s$$

Shift left

Subtract

After *k* iteration $S^{(k)} = 2^k s^{(0)} - q(2^k d) = 2^k [z - (q \times d)] = 2^k s$

Fractional version $S_{frac}^{(j)} = 2s_{frac}^{(j-1)} - q_{-j} d_{frac} \quad with \quad s_{frac}^{(0)} = z_{frac} \quad and \quad s_{frac}^{(k)} = 2^k s_{frac}$

Example of sequential division with integer and fractional operands

Integer division

z		0	1	1	1	0	1	0	1
2^4d		1	0	1	0				
$s^{(0)}$		0	1	1	1	0	1	0	1
$2s^{(0)}$	0	1	1	1	0	1	0	1	
$-q_3 2^4 d$		1	0	1	0	$\{q_3 = 1\}$			
$s^{(1)}$		0	1	0	0	1	0	1	
$2s^{(1)}$	0	1	0	0	1	0	1		
$-q_2 2^4 d$		0	0	0	0	$\{q_2 = 0\}$			
$s^{(2)}$		1	0	0	1	0	1		
$2s^{(2)}$	1	0	0	1	0	1			
$-q_1 2^4 d$		1	0	1	0	$\{q_1 = 1\}$			
$s^{(3)}$		1	0	0	0	1			
$2s^{(3)}$	1	0	0	0	1				
$-q_0 2^4 d$		1	0	1	0	$\{q_0 = 1\}$			
$s^{(4)}$		0	1	1	1				
s						0	1	1	1
q						1	0	1	1

Fractional division

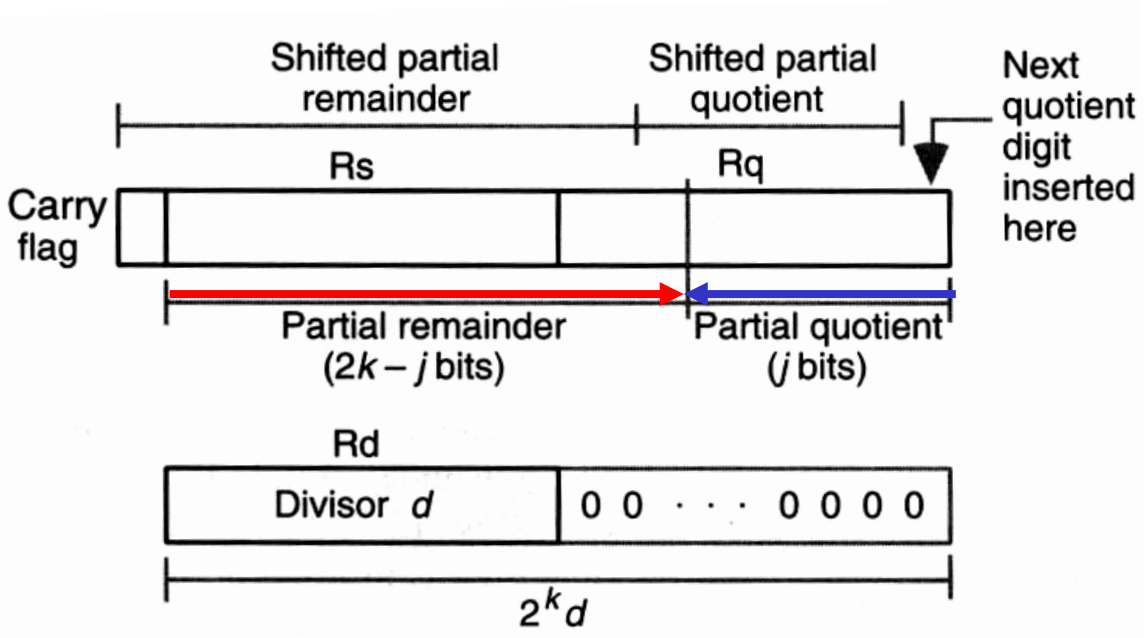
z_{frac}	.	0	1	1	1	0	1	0	1
d_{frac}	.	1	0	1	0				
$s^{(0)}$	.	0	1	1	1	0	1	0	1
$2s^{(0)}$	0	.	1	1	1	0	1	0	1
$-q_{-1}d$		.	1	0	1	0	$\{q_{-1} = 1\}$		
$s^{(1)}$		.	0	1	0	0	1	0	1
$2s^{(1)}$	0	.	1	0	0	1	0	1	
$-q_{-2}d$		.	0	0	0	0	$\{q_{-2} = 0\}$		
$s^{(2)}$		.	1	0	0	1	0	1	
$2s^{(2)}$	1	.	0	0	1	0	1		
$-q_{-3}d$		.	1	0	1	0	$\{q_{-3} = 1\}$		
$s^{(3)}$		.	1	0	0	0	1		
$2s^{(3)}$	1	.	0	0	0	1			
$-q_{-4}d$		.	1	0	1	0	$\{q_{-4} = 1\}$		
$s^{(4)}$		.	0	1	1	1			
s_{frac}		.	0	0	0	0	0	1	1	1
q_{frac}		.	1	0	1	1				

Programmed Division

- Use *shift* and *add* to perform integer division by a processor.
- Two *k*-bit register to store the *partial remainder* and the *quotient*.

Integer division

z		0	1	1	1	0	1	0 1
2^4d		1	0	1	0			
$s^{(0)}$		0	1	1	1	0	1	0 1
$2s^{(0)}$	0	1	1	1	0	1	0	1
$-q_3 2^4 d$		1	0	1	0			$\{q_3 = 1\}$
$s^{(1)}$		0	1	0	0	1	0	1
$2s^{(1)}$	0	1	0	0	1	0	1	
$-q_2 2^4 d$		0	0	0	0			$\{q_2 = 0\}$
$s^{(2)}$		1	0	0	1	0	1	
$2s^{(2)}$	1	0	0	1	0	1		
$-q_1 2^4 d$		1	0	1	0			$\{q_1 = 1\}$
$s^{(3)}$		1	0	0	0	1		
$2s^{(3)}$	1	0	0	0	1			
$-q_0 2^4 d$		1	0	1	0			$\{q_0 = 1\}$
$s^{(4)}$		0	1	1	1			
s					0	1	1	1
q					1	0	1	1



Restoring Hardware Dividers

z			0	1	1	1	0	1	0	1
2^4d	0	1	0	1	0					
-2^4d	1	0	1	1	0					
$s^{(0)}$	0	0	1	1	1	0	1	0	1	
$2s^{(0)}$	0	1	1	1	0	1	0	1		
$+(-2^4d)$	1	0	1	1	0					
$s^{(1)}$	0	0	1	0	0	1	0	1		
$2s^{(1)}$	0	1	0	0	1	0	1			
$+(-2^4d)$	1	0	1	1	0					
$s^{(2)}$	1	1	1	1	1	0	1			
$s^{(2)} = 2s^{(1)}$	0	1	0	0	1	0	1			
$2s^{(2)}$	1	0	0	1	0	1				
$+(-2^4d)$	1	0	1	1	0					
$s^{(3)}$	0	1	0	0	0	1				
$2s^{(3)}$	1	0	0	0	1					
$+(-2^4d)$	1	0	1	1	0					
$s^{(4)}$	0	0	1	1	1					
s						0	1	1	1	
q						1	0	1	1	

restore

unchanged

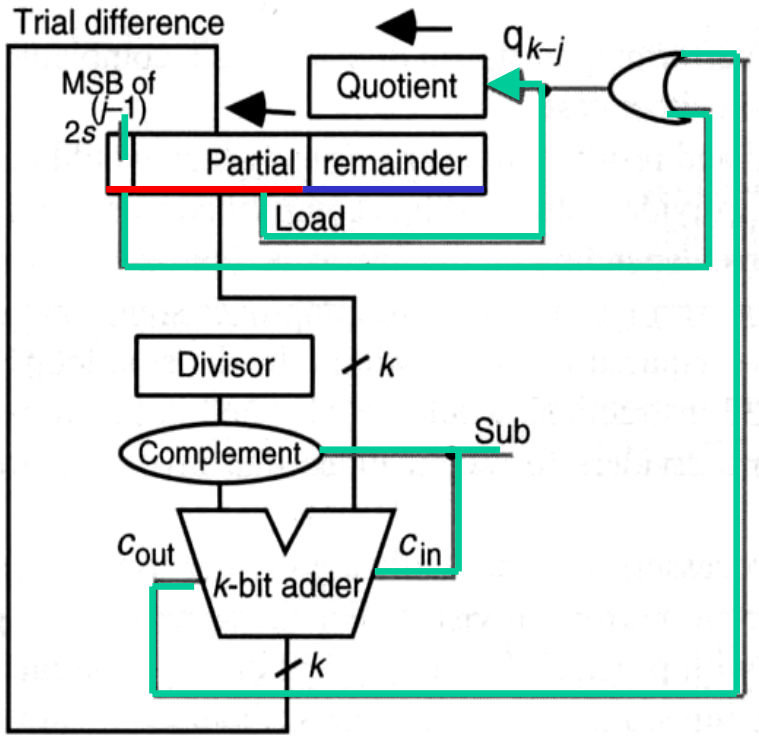
No overflow, since: $(0111)_{two} < (1010)_{two}$

Positive, so set $q_3 = 1$

Negative, so set $q_2 = 0$ and restore

Positive, so set $q_1 = 1$

Positive, so set $q_0 = 1$



Nonrestoring and Signed Division

- Restoring division have timing issues because every cycle must be long enough to allow:
  - Shifting of the registers.
  - Propagation of signals through the adder
  - Storing of the quotient digit.
- To avoid timing issues nonstoring division can be use.
  - Unsigned nonstoring:
    - Skip restore step in restoring division. keeping the incorrect partial product, and add the divisor at next cycle.
  - Signed nonstoring:
    - Avoid correction add operation in unsigned nonstoring by allow quotient to be {-1,1}.

Nonrestoring Unsigned division example

z		0	1	1	1	0	1	0	1
$24d$	0	1	0	1	0				
$-24d$	1	0	1	1	0				
$s^{(0)}$	0	0	1	1	1	0	1	0	1
$2s^{(0)}$	0	1	1	1	0	1	0	1	
$+(-24d)$	1	0	1	1	0				
$s^{(1)}$	0	0	1	0	0	1	0	1	
$2s^{(1)}$	0	1	0	0	1	0	1		
$+(-24d)$	1	0	1	1	0				
$s^{(2)}$	1	1	1	1	1	0	1		
$2s^{(2)}$	1	1	1	1	0	1			
$+24d$	0	1	0	1	0				
$s^{(3)}$	0	1	0	0	0	1			
$2s^{(3)}$	1	0	0	0	1				
$+(-24d)$	1	0	1	1	0				
$s^{(4)}$	0	0	1	1	1				
s						0	1	1	1
q						1	0	1	1

No overflow, since: $(0111)_{two} < (1010)_{two}$

Positive, so subtract

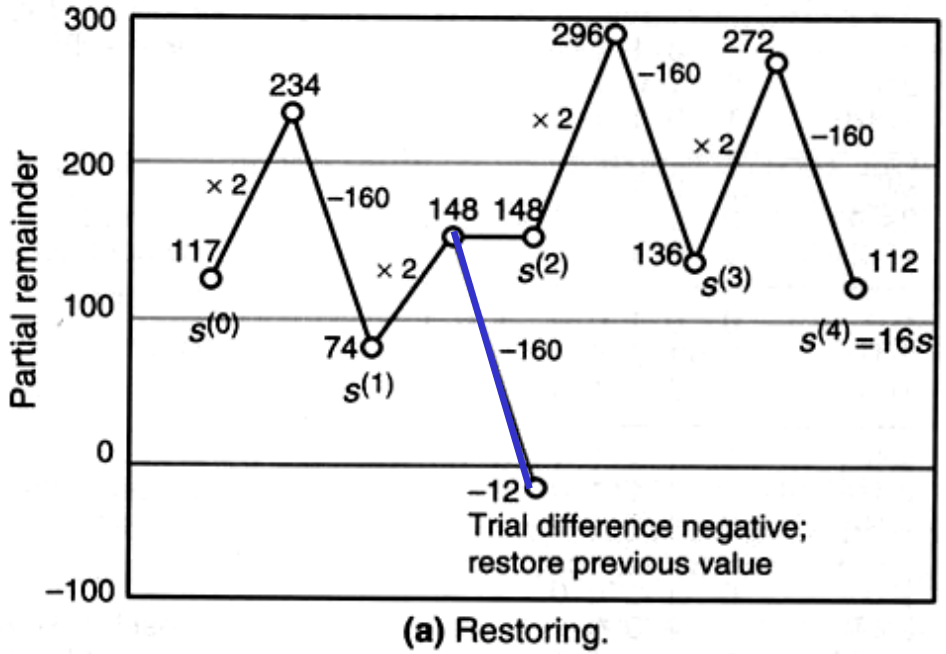
Positive, so set $q_3 = 1$ and subtract

Negative, so set $q_2 = 0$ and add

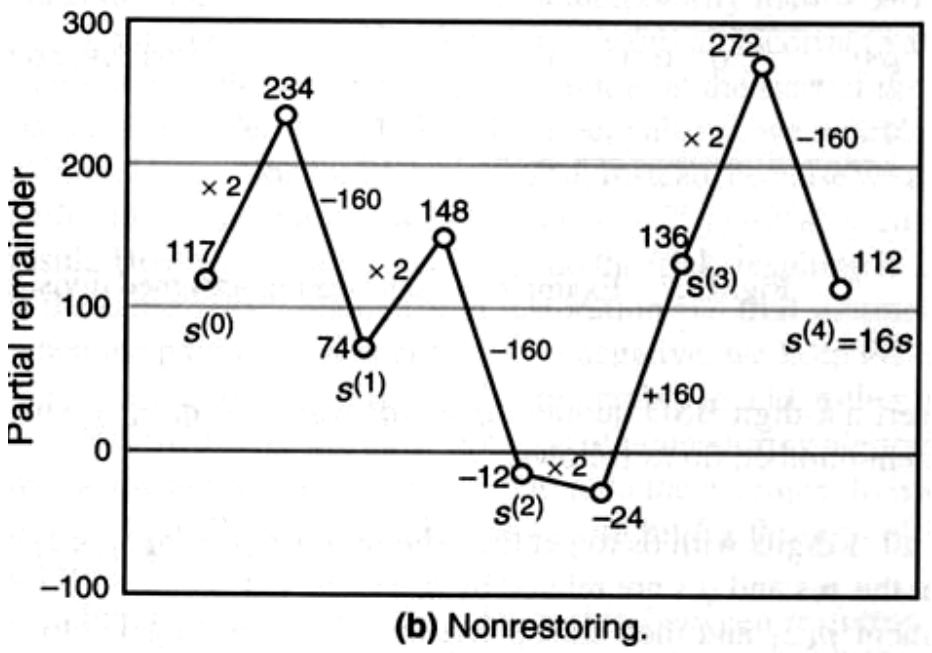
Positive, so set $q_1 = 1$ and subtract

Positive, so set $q_0 = 1$

Partial Remainder variation for restoring and nonrestoring dividsion



(a) Restoring.



(b) Nonrestoring.

Nonstoring Signed Division

- If the quotient digits are selected from the set {1,-1}, (1→sub,-1→add).
- Goal is to end up with a remainder matches the sign of the dividend. (dividend can be positive or negative).
- The rule for quotient digit selection becomes:

$$if\ \ sign(s) = sign(d)\ then\ q_{k-j} = 1\ else\ q_{k-j} = -1$$

Nonstoring Signed Division: Two Problems

- The quotient with digits 1 and -1 must be converted to standard binary.
- If the final remainder *s* has a sign opposite that of *z*, a correction step addition $\pm$ d to the remainder and subtraction of ± 1 from the quotient, is needed.
  - Convert a *k*-digit BSD quotient to a *k*-bit 2's complement number.

    A. replace all -1 digits with 0s to get the k-bit number

    $$p = p_{k-1}p_{k-2}\cdots p_0,\ p_i \in \{0,1\}$$

    B. complement p_{k-1} and then shift *p* left by 1 bit, inserting 1 to the LSB, get

    $$q = (\overline{p}_{k-1}p_{k-2}\cdots p_0 1)_{2's-compl.}$$

Nonstoring Signed Division: example

z		0	0	1	0	0	0	0	1
2^4d	1	1	0	0	1				
-2^4d	0	0	1	1	1				
$s^{(0)}$	0	0	0	1	0	0	0	0	1
$2s^{(0)}$	0	0	1	0	0	0	0	1	
$+2^4d$	1	1	0	0	1				
$s^{(1)}$	1	1	1	0	1	0	0	1	
$2s^{(1)}$	1	1	0	1	0	0	1		
$+(-2^4d)$	0	0	1	1	1				
$s^{(2)}$	0	0	0	0	1	0	1		
$2s^{(2)}$	0	0	0	1	0	1			
$+2^4d$	1	1	0	0	1				
$s^{(3)}$	1	1	0	1	1	1			
$2s^{(3)}$	1	0	1	1	1				
$+(-2^4d)$	0	0	1	1	1				
$s^{(4)}$	1	1	1	1	0				
$+(-2^4d)$	0	0	1	1	1				
$s^{(4)}$	0	0	1	0	1				
s						0	1	0	1
q						$\bar{1}$	1	$\bar{1}$	1
p						0	1	0	1
						/	/	/	/
Shifted p						1	1	0	1	1
$q_{2\text{'s-compl}}$							1	1	0	0

Dividend = $(33)_{ten}$
Divisor = $(-7)_{ten}$

sign($s^{(0)}$) ≠ sign(d), so set $q_3 = \bar{1}$ and add

sign($s^{(1)}$) = sign(d), so set $q_2 = 1$ and subtract

sign($s^{(2)}$) ≠ sign(d), so set $q_1 = \bar{1}$ and add

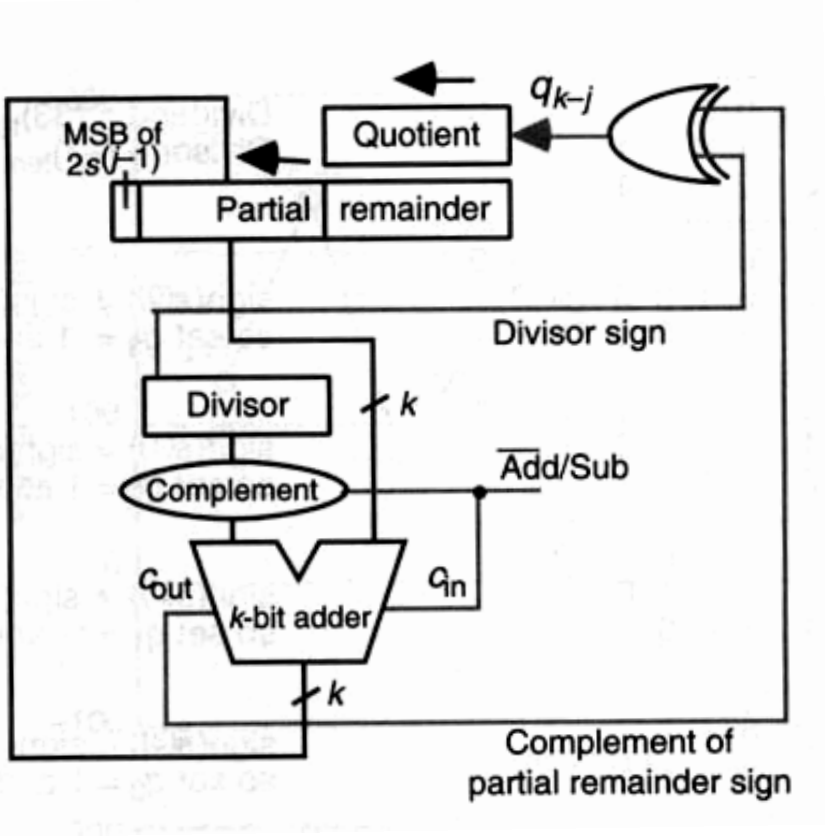
sign($s^{(3)}$) = sign(d), so set $q_0 = 1$ and subtract

sign($s^{(4)}$) ≠ sign(d)
Corrective subtraction

Remainder = $(5)_{ten}$
Ucorrected BSD quotient

$\bar{1}$s replaced by 0s

Add 1 to correct
Quotient = $(-4)_{ten}$

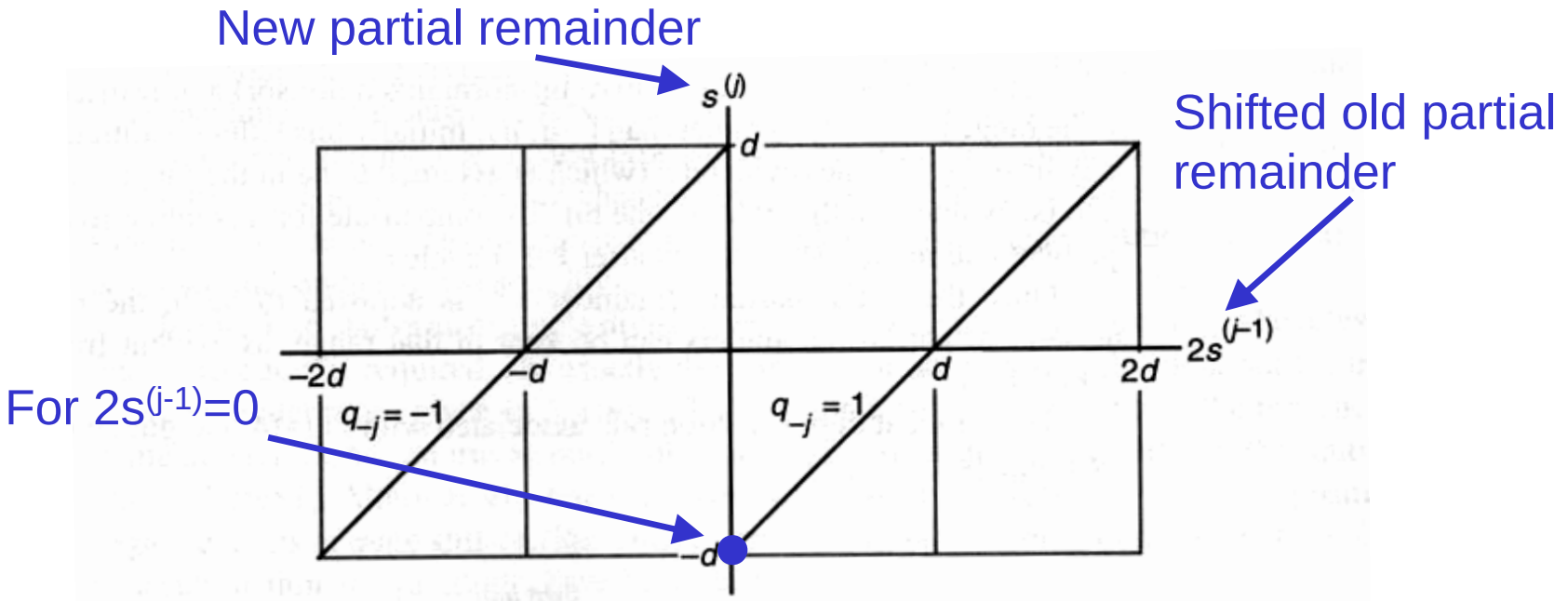


Radix-2 SRT Division: review of nonrestoring division

- Reconsider radix-2 nonrestoring division algorithm for fractional operands.

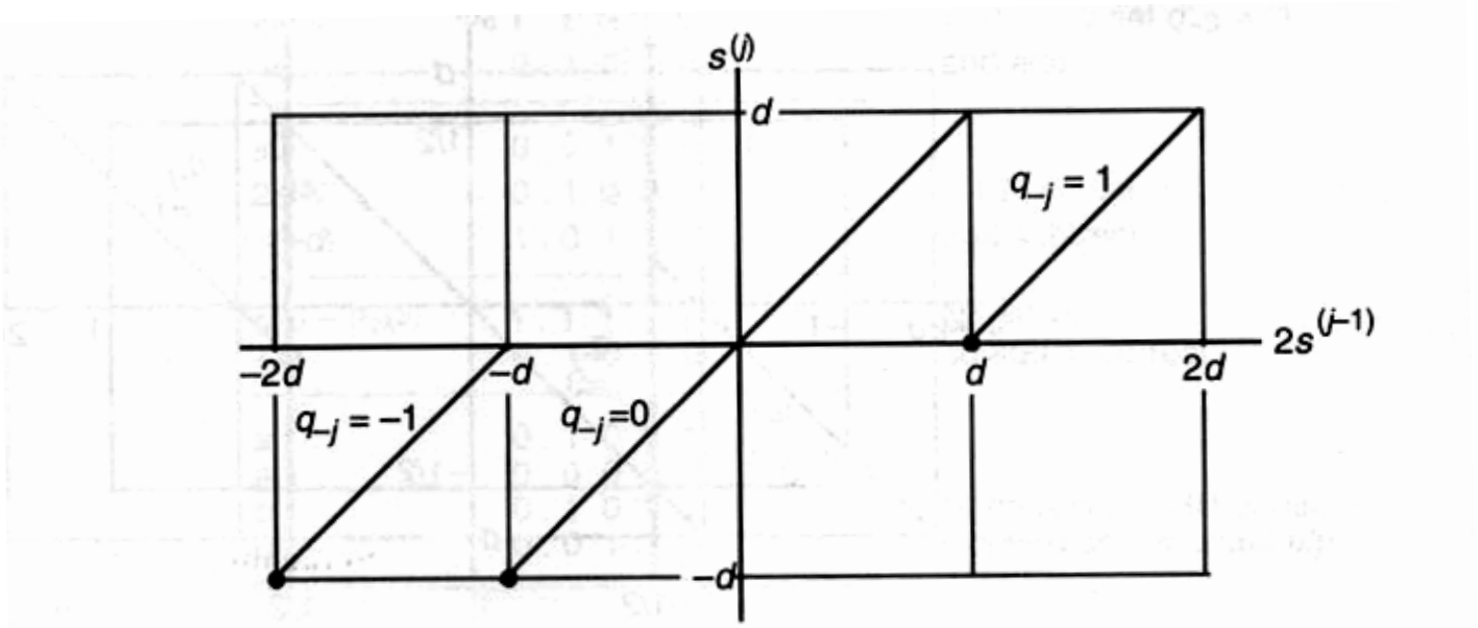
$$S^{(j)} = 2s^{(j-1)} - q_{-j}d \quad with \quad s^{(0)} = z \quad and \quad s^{(k)} = 2^k s$$

- Quotient is obtained with the digit set {-1,1} and is then converted to the standard digit set {0,1}.



Radix-2 SRT Division (con't)

- Quotient is obtained using digit set {-1,0,1}.
- Quotient "0" is selected when q_{-j}=0 for $-d \leq 2s^{(j-1)} < d$
- Quotient "0" is simple shift, can speed up the division operation.
- But determined $-d \leq 2s^{(j-1)} < d$ need trial subtraction. Would consume more time than they save!



Radix-2 SRT Division (con't)

- ***SRT***: *Sweeney*, *Roberson*, and *Tocher* discovered SRT division about the same time.
- *Normalized* divisor and *normalized* partial dividend.
- Divisor and partial dividend is limited in the range [1/2,1) or (-1,-1/2].
- Easier comparison can be used due to normalized divisor.

Radix-2 SRT Division (con't)

- Because of normalized divisor. comparison become:
  - $2s^{(j-1)} > +½ = (0.1)_{2's\text{-compl}}$ implies $2s^{(j-1)} = (0.1u_{-2}u_{-3}\ldots)_{2's\text{-compl}}$
  - $2s^{(j-1)} < -½ = (1.1)_{2's\text{-compl}}$ implies $2s^{(j-1)} = (1.0u_{-2}u_{-3}\ldots)_{2's\text{-compl}}$
- $2s^{(j-1)} > +½$ is given by $u_0\bar{u}_{-1}$, and $2s^{(j-1)} < -½$ is given by . Much easier!.

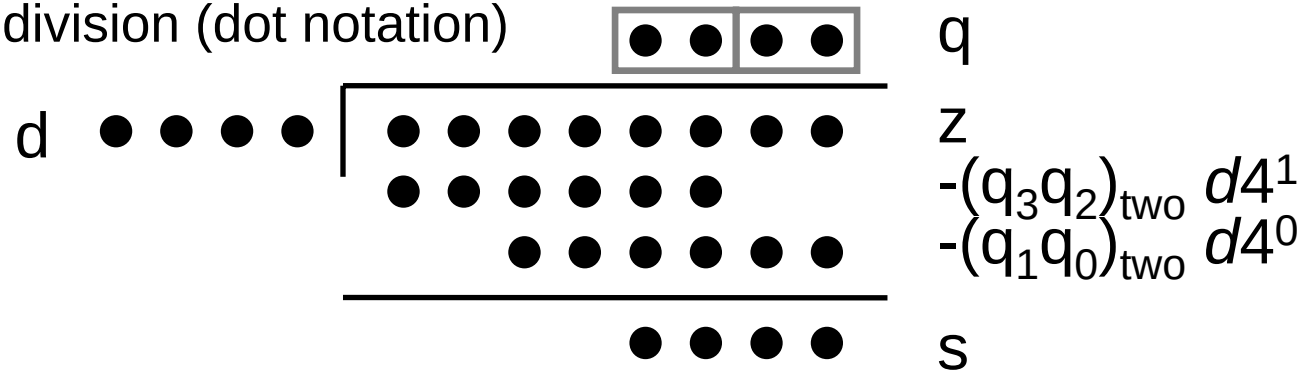
Example of Radix-2 SRT Division

z	.0 1 0 0 0 1 0 1	In [–1/2, 1/2), so OK
d	.1 0 1 0	In [1/2, 1), so OK
$-d$	1.0 1 1 0	
$s^{(0)}$	0.0 1 0 0 0 1 0 1	
$2s^{(0)}$	0.1 0 0 0 1 0 1	≥ 1/2, so set $q_{-1} = 1$
$+(-d)$	1.0 1 1 0	and subtract
$s^{(1)}$	1.1 1 1 0 1 0 1	
$2s^{(1)}$	1.1 1 0 1 0 1	In [–1/2, 1/2), so set $q_{-2} = 0$
$s^{(2)} = 2s^{(1)}$	1.1 1 0 1 0 1	
$2s^{(2)}$	1.1 0 1 0 1	< –1/2, so set $q_{-3} = \bar{1}$
$+d$	0.1 0 1 0	and add
$s^{(3)}$	0.0 1 0 0 1	
$2s^{(3)}$	0.1 0 0 1	≥ 1/2, so set $q_{-4} = 1$
$+(-d)$	1.0 1 1 0	and subtract
$s^{(4)} = 2s^{(3)}$	1.1 1 1 1	Negative, so add to correct
$+d$	0.1 0 1 0	
$s^{(4)}$	0.1 0 0 1	
s	0.0 0 0 0 1 0 0 1	Uncorrected BSD quotient
q	0.1 0 $\bar{1}$ 1	Convert and subtract *ulp*
q	0.0 1 1 0	

Basic of High-Radix Division

z	*Dividend*	$z_{2k-1}z_{2k-2}...z_1z_0$
d	*Divisor*	$d_{k-1}d_{k-2}...d_1d_0$
q	*Quotient*	$q_{k-1}q_{k-2}...q_1q_0$
s	$Remainder[z-(d\times q)]$	$s_{k-1}s_{k-2}...s_1s_0$

Ex: Radix-4 division (dot notation)



High-Radix Division example

Radix-4 integer division

z		0	1	2	3	1	1	2	3
$4^4 d$		1	2	0	3				
$s^{(0)}$		0	1	2	3	1	1	2	3
$4s^{(0)}$	0	1	2	3	1	1	2	3	
$-q_3 4^4 d$	0	1	2	0	3	$\{q_3 = 1\}$			
$s^{(1)}$		0	0	2	2	1	2	3	
$4s^{(1)}$	0	0	2	2	1	2	3		
$-q_2 4^4 d$	0	0	0	0	0	$\{q_2 = 0\}$			
$s^{(2)}$		0	2	2	1	2	3		
$4s^{(2)}$	0	2	2	1	2	3			
$-q_1 4^4 d$	0	1	2	0	3	$\{q_1 = 1\}$			
$s^{(3)}$		1	0	0	3	3			
$4s^{(3)}$	1	0	0	3	3				
$-q_0 4^4 d$	0	3	0	1	2	$\{q_0 = 2\}$			
$s^{(4)}$		1	0	2	1				
s						1	0	2	1
q						1	0	1	2

Radix-10 fractional division

z_{frac}	.	7	0	0	3		
d_{frac}	.	9	9				
$s^{(0)}$	.	7	0	0	3		
$10s^{(0)}$	7	.	0	0	3		
$-q_{-1} d$	6	.	9	3	$\{q_{-1} = 7\}$		
$s^{(1)}$		.	0	7	3		
$10s^{(1)}$	0	.	7	3			
$-q_{-2} d$	0	.	0	0	$\{q_{-2} = 0\}$		
$s^{(2)}$		.	7	3			
s_{frac}		.	0	0	7	3	
q_{frac}		.	7	0			

Features of High-Radix Division

- Dividing binary number in radix 2b reduces the cycles required by a factor of b, but each cycle is more difficult to implement:
  - The higher radix makes the guessing of the correct quotient digit more difficult.
  - The value to be subtracted are determined sequentially, one per cycle. Possible value to be subtracted become harder to generate.
- Other variations in division and divider please consult reference “Computer Arithmetic: algorithm and hardware designs / Behrooz Parhami, OXFORD university press” ch13~ch16.